An interior point algorithm for the general nonlinear programming problem with trust region globalization*

Indraneel Das
Dept. of Computational & Applied Mathematics
Rice University
Houston, Texas-77251

Abstract

This paper presents an SQP-based interior point technique for solving the general nonlinear programming problem using trust region globalization and the Coleman-Li scaling. The SQP subproblem is decomposed into a normal and a reduced tangential subproblem in the tradition of numerous works on equality constrained optimization, and strict feasibility is maintained with respect to the bounds. This is intended to be an extension of previous work by Coleman & Li and Vicente. Though no theoretical proofs of convergence are provided, some computational results are presented which indicate that this algorithm holds promise. The computational experiments have been geared towards improving the semi-local convergence of the algorithm; in particular high sensitivity of the speed of convergence with respect to the fraction of the trust region radius allowed for the normal step and with respect to the initial trust region radius are observed. The chief advantages of this algorithm over primal-dual interior point algorithms are better handling of the 'sticking problem' and a reduction in the number of variables by elimination of the multipliers of bound constraints.

*This research was partially supported by the Dept. of Energy, DOE Grant DE-FG03-95ER25257 and by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

1 Introduction

This paper focuses on an algorithm for solving the general nonlinear programming problem in the form

$$\begin{aligned} &\min_x f(x) \\ s.t.\ \ &h(x) = 0^1 \\ &a \leq x \leq b \end{aligned} \qquad \text{(NLP)}$$

where $f : \mathcal{R}^n \mapsto \mathcal{R}$, $h : \mathcal{R}^n \mapsto \mathcal{R}^m$ are twice continuously differentiable mappings and $m < n$. Recently at least two excellent pieces of work on using trust region globalization in interior point techniques for solving nonlinear optimization problems have appeared. Coleman and Li in [1] and [2] discuss interior point algorithms for solving the bound-constrained problem

$$\begin{aligned} &\min_x f(x) \\ s.t.\ \ &a \leq x \leq b \end{aligned}$$

Vicente in his Ph.D. thesis [3] extensively deals with an extended form of the above problem with equality constraints having a special structure, and bounds only on the control variables u:

$$\begin{aligned} &\min_{y,u} f(y,u) \\ s.t.\ \ &C(y,u) = 0 \\ &a \leq u \leq b \end{aligned}$$

where $dim(C) = dim(y)$.

Both [1] and [3] rigorously prove global convergence, second-order convergence under reasonable assumptions[2] and local q-quadratic convergence in their respective settings. This article draws on the results and accrues on the experiences of the two aforementioned works and extends their algorithms to handle problem (NLP). Though no proofs are provided, the

[1] This takes into account inequality constraints $g(x) \leq 0$ as well since inequalities can be converted to equalities by adding slack variables and imposing nonnegativity bounds on the slacks.

[2] Second order convergence denotes convergence to a point satisfying second order necessary conditions for a local minimum.

algorithm described here has been implemented and computationally tested and found to be reasonably successful; more testing is in progress. An important observation made in this paper is that the trust region subproblems in the scaled and unscaled steps are not equivalent, contrary to Coleman & Li's claim in [1], and this fact is proved in the appendix.

2 Preliminaries

The proposed algorithm here, like its predecessors in Coleman and Li [1] and Vicente [3], starts at a point strictly feasible with respect to the bounds on the variables and produces iterates that are strictly feasible with respect to the bounds (i.e. 'in the interior'). The steps in the algorithm, as in the above two, can be motivated by applying Newton's method to a very special statement of neccesary condition for optimality for problem (NLP), as stated in the following exposition:

Let $l(x, \lambda) = f(x) + \lambda^T h(x)$ (which is *not* the Lagrangian for problem (NLP), since the Lagrangian would also involve the multipliers corresponding to the bound constraints). Define the diagonal scaling matrix $D(x)$ as

$$D(x) = diag(d(x)),$$

where

$$d_i(x) = \begin{cases} \sqrt{b_i - x_i}, & \text{if } (\nabla_x l(x,\lambda))_i < 0 \text{ and } b_i < \infty \\ \sqrt{x_i - a_i}, & \text{if } (\nabla_x l(x,\lambda))_i \geq 0 \text{ and } a_i > -\infty, \\ 1, & \text{otherwise} \end{cases}$$

Then (x^*, λ^*) satisfy first order necessary conditions for optimality of problem (NLP) if and only if

$$D^2(x^*)\nabla_x l(x^*, \lambda^*) = 0$$

$$h(x^*) = 0 \tag{1}$$

The proof for the above becomes obvious on a closer scrutiny of the Karush-Kuhn-Tucker conditions for problem (NLP), which requires the existence of $(x^*, \lambda^*, \mu_a^*, \mu_b^*)$ satisfying

$$\nabla_x l(x^*, \lambda^*) - \mu_a^* + \mu_b^* = 0$$

$$h(x^*)=0 \quad ,$$

where μ_a^* and μ_b^* are vectors of non-negative multipliers of the finite bound constraints, satisfying the complementarity conditions:

$$\mu_{a_i}^*(a_i - x_i^*) = 0$$

$$\mu_{b_i}^*(b_i - x_i^*) = 0 \tag{2}$$

for all i corresponding to x_i with finite bounds.

Keeping in mind that a positive or negative value of $(\nabla_x l(x^*, \lambda^*))_i$ can only be canceled in the above by respectively a positive μ_{a_i} or a positive μ_{b_i}, complementarity makes the equivalence between (1) and (2) obvious.

Quite clearly, other possible choices of $D(x)$ exist which permit the above equivalence, however the above choice with square roots of the distances of the variables from the bounds permits local q-quadratic convergence for the treatments in Coleman and Li[1] and Vicente [3] in spite of the nonsmoothness of $D^2(x)$ and hence is our choice here.

2.1 Newton step

Let $\eta(x)$ be a vector such that

$$\eta_i(x) = \frac{\partial(d_i^2(x))}{\partial x_i}, i = 1, \ldots, n$$

The above definition of $\eta(x)$ is equivalent to

$$\eta_i(x) = \begin{cases} -1, & \text{if } (\nabla_x l(x,\lambda))_i < 0 \text{ and } b_i < \infty \\ 1, & \text{if } (\nabla_x l(x,\lambda))_i \geq 0 \text{ and } a_i > -\infty, \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Then it can be shown that a Newton step in (x, λ) on nonlinear system (1) is given by

$$\begin{aligned} [D^2(x)\nabla_x^2 l(x,\lambda) + diag(\nabla_x l(x,\lambda))diag(\eta(x))]\triangle x + D^2(x)\nabla_x h(x)\triangle\lambda \\ = -D^2(x)\nabla_x l(x,\lambda) \\ \nabla_x h(x)^T \triangle x = -h(x) \end{aligned} \tag{4}$$

Multiplying throughout on the left of the first equation by $D^{-2}(x)$, the above system can be equivalently written as

$$\begin{aligned}[\nabla_x^2 l(x,\lambda)+diag(\nabla_x l(x,\lambda))diag(\eta(x))D^{-2}(x)]\triangle x+\nabla_x h(x)\triangle\lambda &= -\nabla_x l(x,\lambda)\\ \nabla_x h(x)^T\triangle x &= -h(x)\end{aligned} \tag{5}$$

Introducing the scaled step $\triangle\hat{x} = D^{-1}(x)\triangle x$, the above system takes the form

$$\begin{aligned}[\nabla_x^2 l(x,\lambda)D(x) + diag(\nabla_x l(x,\lambda))diag(\eta(x))D^{-1}(x)]\triangle\hat{x} + \nabla_x h(x)\triangle\lambda &\\ = -D^2(x)\nabla_x l(x,\lambda)&\\ [D(x)\nabla_x h(x)]^T\triangle\hat{x} = -h(x)&\end{aligned}$$

Multiplying throughout on the left of the first equation by $D(x)$ and remembering that diagonal matrix multiplication commutes, we arrive at the system

$$\begin{aligned}[(D(x)\nabla_x^2 l(x,\lambda)D(x) + diag(\nabla_x l(x,\lambda))diag(\eta(x))]\triangle\hat{x} + D(x)\nabla_x h(x)\triangle\lambda &\\ = -D(x)\nabla_x l(x,\lambda)&\\ [D(x)\nabla_x h(x)]^T\triangle\hat{x} = -h(x)&\end{aligned} \tag{6}$$

3 Trust region SQP formulation

The trust region subproblems based on the above Newton steps can be formulated in terms of the scaled step $\hat{s}$ or in terms of the unscaled step; we shall do both and compare their performances. Even though Coleman & Li in [1], pg. 422, state that the two formulations are equivalent, we prove that the steps generated by the two formulations can be completely different when the Newton step does not lie within the trust region (see Appendix A). In fact, this may be the only explanation for their difference in computational performances. Even though the scaling is only a linear transformation, it is introduced in a nonlinear setting, and hence creates a difference.[3]

[3]This is similar to the phenomenon that the steps generated by the log barrier method in primal-dual interior point techniques and those generated by the damped, perturbed KKT formulation never agree, as shown by Tapia (see Tech Report TR92-40, Dept. of Computational & Applied Math, Rice University by El-Bakry, Tapia, Zhang and Tsuchiya).

3.1 Scaled step subproblems

System (6) suggests the following trust-region SQP [4] subproblem in terms of the scaled step $\hat{s}$ ($\|.\|$ denotes the l_2 norm all throughout):

$$\begin{aligned} \min_{\hat{s}} \quad & \frac{1}{2}\hat{s}^T H \hat{s} + [D(x)\nabla_x l(x,\lambda)]^T \hat{s} \\ s.t. \quad & [D(x)\nabla_x h(x)]^T \hat{s} + h(x) = 0 \\ & \|\hat{s}\| \leq \delta \end{aligned}$$

where

$$H = D(x)\nabla_x^2 l(x,\lambda)D(x) + diag(\nabla_x l(x,\lambda))diag(\eta(x))$$

It is well-known that if the trust region constraint on the length of the step $\hat{s}$ is not active, the solution to the above in the primal and dual variables is exactly the Newton step in $(\hat{x},\lambda)$ given by system (6). However, this subproblem may well be infeasible, and this is taken care of by a bilevel step decomposition approach, the description of which follows.

3.1.1 Step decomposition

In the tradition of numerous works on equality constrained optimization works in recent years (see, for example, [8], [10], [4], [3], to name only a few), we adopt a bilevel approach and decompose the trust region SQP subproblem above into the two subproblems below:

$$\begin{aligned} \min_{s_n} \quad & \|[D(x)\nabla_x h(x)]^T s_n + h(x)\|^2 \\ s.t. \quad & \|s_n\| \leq \nu\delta, \end{aligned}$$

where $\nu \in (0,1)$, usually chosen to be $\in [0.5, 0.8]$, followed by

$$\begin{aligned} \min_{s_t} \quad & \frac{1}{2}\hat{s}^T H \hat{s} + [D(x)\nabla_x l(x,\lambda)]^T \hat{s} \\ s.t. \quad & [D(x)\nabla_x h(x)]^T \hat{s} = [D(x)\nabla_x h(x)]^T s_n \\ & \|\hat{s}\| \leq \delta, \end{aligned}$$

[4] Sequential Quadratic Programming

where $\hat{s}$ is the full step $\hat{s} = s_t + s_n$. The linear equality constraints ensure that the improvement in the model for attaining feasibility is not disturbed. The latter subproblem can be written in terms of step s_t in the more useful form below:

$$\begin{aligned} \min_{s_t} \quad & \frac{1}{2} s_t^T H s_t + [D(x)\nabla_x l(x,\lambda) + H s_n]^T s_t \\ s.t. \quad & [D(x)\nabla_x h(x)]^T s_t = 0 \\ & \|s_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}. \end{aligned}$$

Then x is updated as

$$x \leftarrow x + D(x)(s_t + s_n).$$

The first of these is usually called the normal or vertical subproblem, while the latter is referred to as the tangential or horizontal subproblem. It is again clear that if the trust-region constraint is inactive in both the subproblems, then the step $s_n + s_t$ is exactly the same as the Newton step in $\hat{x}$ given by system (6). If the trust region radius is too small to allow the earlier 'undecomposed' subproblem to be feasible, the normal subproblem has the trust region constraint active. Observe how not having $[D(x)\nabla_x h(x)]^T s_n + h(x) = 0$, i.e., not having the earlier 'undecomposed' subproblem feasible, does not affect the algorithm any more.

It is important to note here that this bilevel aproach does not actually give us a step in λ. So we update lambda using a projection formula. Remembering that at optimality

$$D^2(x^*)\nabla_x l(x^*, \lambda^*) = 0$$

$$i.e., \quad D^2(x^*)\nabla_x f(x^*) + D^2(x^*)\nabla_x h(x^*)\lambda^* = 0$$

we take λ at each iteration to be the least squares solution of

$$D^2(x)\nabla_x h(x)\lambda = -D^2(x)\nabla_x f(x)$$

or, in other words,

$$\lambda = -[(D^2(x)\nabla_x h(x))^T (D^2(x)\nabla_x h(x))]^{-1} (D^2(x)\nabla_x h(x))^T D^2(x)\nabla_x f(x). \tag{7}$$

This computation is elaborated on later.

Following common practice in equality constrained optimization, we do not actually solve the tangential subproblem with the equality constraints $[D(x)\nabla_x h(x)]^T s_t = 0$ but instead force the step s_t to be in $\mathcal{N}([D(x)\nabla_x h(x)]^T)$, the nullspace of the scaled Jacobian of $h(x)$ so that the equality constraints are automatically satisfied. More specifically, the QR factorization of $[D^2(x)\nabla_x h(x)]^T$ is computed, which also helps with the multiplier update, and a basis $Z(x)$ for the nullspace of $[D^2(x)\nabla_x h(x)]^T$ is extracted from the last $n-m$ columns of Q. Note that $[D^2(x)\nabla_x h(x)]^T$,$[D(x)\nabla_x h(x)]^T$ and $\nabla_x h(x)^T$ have the same nullspace, since $a < x < b$ makes $D(x)$ necessarily nonsingular. Theoretical details on this nullspace approach including the issue of the differentiablity of the matrix $Z(x)$ can be found in Goodman [7] .

Once $Z(x)$ is obtained, the step s_t is forced to satisfy

$$s_t = Z(x)\bar{s}_t.$$

Substituting this in the tangential subproblem described earlier yields

$$\min_{\bar{s}_t} \quad \frac{1}{2}\bar{s}_t^T Z(x)^T H Z(x)\bar{s}_t + [Z(x)^T(D(x)\nabla_x l(x,\lambda) + Hs_n)]^T\bar{s}_t$$

$$s.t. \quad \|Z(x)\bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}.$$

Since the columns of $Z(x)$ are orthonormal, $\|Z(x)\bar{s}_t\| = \|\bar{s}_t\|$ [5], yielding the following problem which we shall henceforth refer to as the reduced tangential subproblem

$$\min_{\bar{s}_t} \quad \frac{1}{2}\bar{s}_t^T Z(x)^T H Z(x)\bar{s}_t + [Z(x)^T(D(x)\nabla_x l(x,\lambda) + Hs_n)]^T\bar{s}_t$$

$$s.t. \quad \|\bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}.$$

Now x is updated as $x \leftarrow x + s$, where

$$s = D(x)(s_n + Z(x)\bar{s}_t).$$

[5] $s_t^T Z^T Z s_t = s_t^T s_t$, since $Z^T Z$ is the $n-m \times n-m$ identity.

3.2 Unscaled step subproblems

The trust region subproblem in terms of the unscaled step can be derived in one of three ways (all of which may not follow the formulation of trust-region theory):

Based on the Newton steps directly

Based on the Newton step in (5), the trust region subproblem in the unscaled step can be written directly as

$$\min_s \frac{1}{2} s^T \tilde{H} s + \nabla_x l(x,\lambda)^T s$$

$$s.t. \qquad \nabla_x h(x)^T s + h(x) = 0$$

$$\|s\| \leq \delta,$$

where

$$\tilde{H} = \nabla_x^2 l(x,\lambda) + diag(\nabla_x l(x,\lambda)) diag(\eta(x)) D^{-2}(x).$$

Decomposing the step s as before, the normal and tangential subproblems, respectively, turn out to be

$$\min_{s_n} \qquad \|\nabla_x h(x)^T s_n + h(x)\|^2$$

$$s.t. \qquad \|s_n\| \leq \nu\delta$$

and

$$\min_{\bar{s}_t} \qquad \frac{1}{2} \bar{s}_t^T Z(x)^T \tilde{H} Z(x) \bar{s}_t + [Z(x)^T (\nabla_x l(x,\lambda) + \tilde{H} s_n)]^T \bar{s}_t$$

$$s.t. \qquad \|\bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}$$

Then the full step s becomes

$$s = s_n + Z(x)\bar{s}_t$$

This formulation will generally be referred to as **unscaled version 1**

The untraditional and possibly unwanted feature of this is that the negative gradient step for the horizontal subproblem is along $-\nabla_x l(x,\lambda)$ and not along $-D^2(x)\nabla_x l(x,\lambda)$, which is what the KKT conditions require. This is dealt with in the following formulation.

Based on the original Newton step in (4), only the horizontal subproblem turns out a little different:

$$\min_{\bar{s}_t} \quad \frac{1}{2}\bar{s}_t^T Z(x)^T D^2(x)\tilde{H} Z(x)\bar{s}_t + [Z(x)^T D^2(x)(\nabla_x l(x,\lambda) + \tilde{H} s_n)]^T \bar{s}_t$$

$$s.t. \quad \|\bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}.$$

The negative gradient step now does point along $-D^2(x)\nabla_x l(x,\lambda)$, however the symmetricity of the second order term is lost, which is also untraditional in trust region formulations.

This formulation will be referred to as **unscaled version 2**

Restoring the scale in the scaled step subproblems

Yet a third formulation can be obtained indirectly by substituting

$$s_n \leftarrow D^{-1}(x)s_n$$

$$\text{and } Z\bar{s}_t \leftarrow D^{-1}(x)Z\bar{s}_t$$

in the scaled step subproblems. The normal and tangential subproblems thus obtained are:

$$\min_{s_n} \quad \|\nabla_x h(x)^T s_n + h(x)\|^2$$

$$s.t. \quad \|D^{-1}(x)s_n\| \leq \nu\delta$$

and

$$\min_{\bar{s}_t} \quad \frac{1}{2}\bar{s}_t^T Z(x)^T D^{-1}(x) H D^{-1}(x) Z(x)\bar{s}_t + [Z(x)^T(\nabla_x l(x,\lambda) + D^{-1} H s_n)]^T \bar{s}_t$$

$$s.t. \quad \|D^{-1}(x)Z\bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2},$$

where, as defined before,

$$H = D(x)\nabla_x^2 l(x,\lambda)D(x) + diag(\nabla_x l(x,\lambda))diag(\eta(x)).$$

Also observe that

$$D^{-1}(x)H D^{-1}(x) = \tilde{H}.$$

This formulation is along the lines of Coleman & Li and it is a formulation of this type that they use to prove their convergence results. However, this

is *not* equivalent to the scaled step subproblem in the sense that the steps obtained may not be identical when the trust region constraint is binding (see Appendix A).

This formulation will be referred to as **unscaled version 3.**

Mixed unscaled version

An observation is in order regarding the vertical subproblem: if $\hat{s}_n$ is the Newton step for the vertical subproblem in the scaled step and s_n is the Newton step for vertical subproblem in unscaled versions 1 or 2, then in general,

$$D(x)\hat{s}_n \neq s_n.$$

Proof:

The scaled Newton step satisfies

$$[D(x)\nabla_x h(x)]^T \hat{s}_n = -h(x);$$

i.e.,

$$\nabla_x h(x)^T (D(x)\hat{s}_n) = -h(x).$$

The unscaled (versions 1 and 2) Newton steps satisfy

$$\nabla_x h(x)^T s_n = -h(x).$$

The preceding yield

$$\nabla_x h(x)^T (D(x)\hat{s}_n - s_n) = 0,$$

which necessarily implies $D(x)\hat{s}_n - s_n = 0$ only if $\nabla_x h(x)^T$ has full column rank, which is never true because in our very problem statement we have $m < n$ and $\nabla_x h(x)^T \in \mathcal{R}^{m \times n}$.
(QED)

Our computational experiments suggest that this fact could be responsible for poor performance of the scaled version (or even unscaled version 3, which is derived from the scaled version) in some problems, which motivates the following *mixed unscaled version.* This version has the normal subproblem as in unscaled version 1 (or 2) and the tangential subproblem as in unscaled version 3, i.e.,

$$\min_{s_n} \quad \|\nabla_x h(x)^T s_n + h(x)\|^2$$

$$s.t. \quad \|s_n\| \leq \nu\delta,$$

followed by

$$\min_{\bar{s}_t} \quad \frac{1}{2}\bar{s}_t^T Z(x)^T D^{-1}(x) H D^{-1}(x) Z(x)\bar{s}_t + [Z(x)^T(\nabla_x l(x,\lambda) + D^{-1} H s_n)]^T \bar{s}_t$$

$$s.t. \quad \|D^{-1}(x) Z \bar{s}_t\| \leq \sqrt{\delta^2 - \|s_n\|^2}.$$

3.3 Interiorization

After calculating the step s, it is scaled by the damping parameter τ to ensure that the updated x stays strictly feasible with respect to the bounds. More precisely, let us define

$$u_i = \begin{cases} \frac{a_i - x_i}{s_i}, & \text{if } a_i > -\infty \text{ and } s_i < 0 \\ 1, & \text{otherwise} \end{cases}$$

$$v_i = \begin{cases} \frac{b_i - x_i}{s_i}, & \text{if } b_i < \infty \text{ and } s_i > 0 \\ 1, & \text{otherwise} \end{cases}$$

Then

$$\tau = \sigma \min(1, \min_i\{u_i, v_i\}), \tag{8}$$

where $\sigma = 0.99995$ forces strict feasibility of x with respect to the bounds.

However, if some x_i is too close to a bound and if the corresponding s_i violates that bound, it could happen that the corresponding u_i or v_i is too small, making the damping parameter very small, resulting in very short steps. To prevent this from happening, we choose a tolerance κ and impose the following conditions:

$$\text{if } u_i < \kappa, \quad s_i = 0, \quad u_i \leftarrow 1$$

$$\text{if } v_i < \kappa, \quad s_i = 0, \quad v_i \leftarrow 1.$$

In our implementation, we chose $\kappa = 10^{-8}$. Thus the interiorized step is

$$s \leftarrow \tau s.$$

3.4 Step acceptance criterion

As is common in trust region algorithms, whether a step is accepted or not is based on the accuracy of the actual decrease in a merit function relative to the predicted decrease. Let us denote by $\bar{l}$ the commonly chosen l_2 norm augmented Lagrangian merit function in equality constrained optimization:

$$\bar{l}(x, \lambda, \rho) = f(x) + \lambda^T h(x) + \rho \|h(x)\|^2,$$

where ρ is a positive number bounded below and is updated at each iteration. Most standard convergence results require $\{\rho_k\}$ to form a nondecreasing sequence. However, El-Alem [5], [6] has proved convergence for a nonmonotonic update of ρ.

Let $x^+ = x + s$, λ^+ be the λ updated according to the projection formula (7). Adapting from Coleman and Li [1], we define actual reduction as

$$ared = \bar{l}(x, \lambda, \rho) - \bar{l}(x^+, \lambda^+, \rho) - \frac{1}{2} s^T [diag(\nabla_x l(x, \lambda)) diag(\eta(x)) D^{-2}(x)] s.$$

The last extra term accounts for the fact that $\bar{l}$ is not exactly the augmented Lagrangian for our general NLP, which also has bounds on x. It should be noted that Vicente in [3] proves all his results without this term in his expression for $ared$.

Recalling the Newton step in (5) on the original KKT system, the predicted decrease in the quadratic model is expressed as

$$pred = -[\nabla_x l(x, \lambda)]^T s - \frac{1}{2} s^T [\nabla_x^2 l(x, \lambda) + diag(\nabla_x l(x, \lambda)) diag(\eta) D^{-2}(x)] s -$$

$$(\nabla_x h(x)^T s + h(x))^T \triangle\lambda + \rho(\|h(x)\|^2 - \|\nabla_x h(x)^T s + h(x)\|^2)$$

$$\text{where} \triangle\lambda = \lambda^+ - \lambda.$$

The step is usually rejected when $\frac{ared}{pred} < 10^{-4}$, and accepted otherwise. However, computational experience shows that approximate solutions to the trust region subproblems can often result in negative values of $pred$. In such a case the step could be admitted even if $ared < 0$. To prevent this, our implementation requires $ared > 0$ in addition to $\frac{ared}{pred} \geq 10^{-4}$ for step acceptance. The trust region radius is updated according to a rule based on $\frac{ared}{pred}$

described later.

A startling observation made in course of our computational experiments was that the steps generated by our algorithm seemed to alternate (we don't necessarily mean exactly every other step) between improving either the merit function $\bar{l}(x, \lambda, \rho)$ or the merit function defined by

$$\phi(x, \lambda) = \left\| \begin{array}{c} D^2(x)\nabla_x l(x, \lambda) \\ h(x) \end{array} \right\|_2^2,$$

which is a reasonable merit function considering that

$$\left[\begin{array}{c} D^2(x)\nabla_x l(x, \lambda) \\ h(x) \end{array} \right] = 0$$

is the system of nonlinear equations which we aim to solve. Computational observations suggest that except in the neighborhood of the solution, both the merit functions rarely exhibit improvement simultaneously. So using only the $\bar{l}(x, \lambda, \rho)$ merit function could lead to rejection of steps which actualy make progress towards the solution. In our implementation, we declare a step as acceptable when it satisfies $\frac{ared}{pred} > 10^{-4}$ for any one of the two merit functions. This strategy results in a huge reduction in the number of iterations, and hence function and gradient evaluations. Actual reduction for $\phi(x, \lambda)$ is obviously

$$ared_\phi = \phi(x, \lambda) - \phi(x^+, \lambda^+).$$

Remembering that

$$\phi(x, \lambda) = \left\| \begin{array}{c} D^2(x)\nabla_x l(x, \lambda) \\ h(x) \end{array} \right\|_2^2 = \|D^2(x)\nabla_x l(x, \lambda)\|^2 + \|h(x)\|^2,$$

predicted decrease can be defined as

$$pred_\phi^{(1)} = \phi(x, \lambda) - \|H_0 s + D^2(x)\nabla_x l(x, \lambda)\|^2 - \|\nabla_x h(x)^T s + h(x)\|^2,$$

where

$$H_0 = D^2(x)\nabla_x^2 l(x, \lambda) + diag(\nabla_x l(x, \lambda))diag(\eta(x)).$$

Alternately, since

$$\nabla_{x,\lambda} \left[\begin{array}{c} D^2(x)\nabla_x l(x, \lambda) \\ h(x) \end{array} \right] = \left[\begin{array}{cc} H_0 & \nabla_x h(x) \\ \nabla_x h(x)^T & 0 \end{array} \right]$$

an expression for pred (using only a first order approximation) could be

$$pred_\phi^{(2)} = -2 \left[\begin{array}{cc} D^2(x)\nabla_x l(x,\lambda) & h(x) \end{array} \right] \left[\begin{array}{cc} H_0 & \nabla_x h(x) \\ \nabla_x h(x)^T & 0 \end{array} \right] \left[\begin{array}{c} s \\ \triangle\lambda \end{array} \right].$$

In our computation, we take $pred_\phi = pred_\phi^{(1)}$, unless $pred_\phi^{(1)} < 0$, in which case we take $pred_\phi = \max\{pred_\phi^{(1)}, pred_\phi^{(2)}\}$.

4 Further Computational Details

4.1 Solving the subproblems

There has been a considerable amount of work on computing approximate solutions of trust region subproblems in recent years (see for example More & Sorensen [9], Zhang [11], Santos & Sorensen [12], Sorensen [13], Steihaug [14]). Standard convergence results usually require the computed step to satisfy a fraction of Cauchy decrease or a fraction of optimal decrease in the merit function, the latter being computationally more expensive but providing stronger results. Introductory material on this can be found in Dennis & Schnabel [15].

Our forthcoming implementation uses More and Sorensen's subroutine DGQTPAR described in [9] to compute the solution for the reduced tangential subproblem. The chief advantage of this is that it mostly returns a step satisfying fraction of optimal decrease even if the subproblem is nonconvex. Since it might be unreasonable to expect a positive definite $Z(x)^T H Z(x)$[6] this property of DGQTPAR is very desirable.

The normal subproblem in the scaled step can be shown to be equivalent to

$$\min_{s_n} \frac{1}{2} s_n^T \mathcal{H} s_n + \gamma^T s_n$$
$$s.t. \|s_n\| \leq \nu\delta,$$

where

$$\mathcal{H} = [D(x)\nabla_x h(x)][D(x)\nabla_x h(x)]^T,$$
$$\gamma = D(x)\nabla_x h(x) h(x).$$

[6]Since there is no guarantee that $(\nabla_x l(x^+,\lambda^+) - \nabla_x l(x,\lambda^+))^T s > 0$, not even the BFGS update is guaranteed to be positive definite.

This subproblem is also solved using DGQTPAR. It should be noted that the matrix $\mathcal{H}$ is positive semidefinite and is necessarily singular; however DGQTPAR can handle this singularity.

The normal subproblem in the unscaled step is exactly the same as the above with $D(x)$ replaced by I_n.

It must be mentioned that the computational results reported later were generated using a Matlab implementation, which approximates the solutions to the above trust region subproblems using dogleg steps (see, for example, Dennis & Schnabel [15]).

4.2 Role of QR factorization

The QR factorization of $D^2(x)\nabla_x h(x)$ plays an important role in our algorithm and implementation and hence deserves special mention. Let the QR factorization for $D^2(x)\nabla_x h(x)$ be partitioned as below[7] (recall that $m = dim(h)$)

$$Y = Q(:, 1:m)$$

$$Z = Q(:, m+1:n)$$

$$\bar{R} = R(1:m, 1:m).$$

The columns of Y contain an orthonormal basis for the range of $D^2(x)\nabla_x h(x)$, or of $\nabla_x h(x)$, since $D^2(x)$ is nonsingular when $a < x < b$, and the columns of Z contain an orthonormal basis for the nullspace of $[D^2(x)\nabla_x h(x)]^T$ or of $\nabla_x h(x)^T$. Further, $\bar{R}$ is upper triangular and nonsingular and rows $m+1:n$ of R contain only zeros. Hence

$$QR = Y\bar{R}$$

The use of Z in eliminating the equality constraints in the tangential subproblem has already been discussed; now we discuss how the QR factorization is used in updating λ using projection formula (7) and in computing s_n.

Substituting $D^2(x)\nabla_x h(x) = QR = Y\bar{R}$, projection formula (7) takes the form

$$\lambda = -[\bar{R}^T Y^T Y \bar{R}]^{-1} \bar{R}^T Y^T D^2(x)\nabla_x f(x)$$

[7] We assume that the reader is familiar with Matlab or FORTRAN 90 syntax for referencing matrices.

$$= -[\bar{R}^T \bar{R}]^{-1} \bar{R}^T Y^T D^2(x) \nabla_x f(x)$$

$$\Longleftrightarrow$$

$$\bar{R}^T \bar{R} \lambda = -\bar{R}^T Y^T D^2(x) \nabla_x f(x).$$

Since $\bar{R}^T$ is nonsingular, the above is equivalent to

$$\bar{R} \lambda = -Y^T D^2(x) \nabla_x f(x)$$

Since $\bar{R}$ is upper triangular, λ can be found from the above by simple back substitution involving only $O(m^2)$ operations, which is considerably economic in comparison with standard Gaussian elimination on (7).

An important observation here is that the computation of λ^+ is dependent on $D(x^+)$, which itself depends on the sign of $(\nabla_x l(x^+, \lambda^+))_i$, and thus requires an estimate of λ^+. In practice, $D(x)$ is updated first and $D(x^+)$ is obtained using the old multipliers; i.e., $D(x^+)$ is obtained using the signs of the components of $\nabla_x l(x^+, \lambda^{(i)})$. The QR factorization of $D(x^+)\nabla_x h(x^+)$ is then computed and stored for use in the next iteration and λ^+ is obtained using this new QR factorization as in (7). This intermediate QR factorization can be an unnecessary additional expense when steps get rejected frequently, but can probably be reused in the next step wherever the model is good, in particular in the neighborhood of a solution.

However, in practice, even this strategy can perform poorly and give inaccurate results, and we need yet another round of predictor-corrector refinement of $D(x)$ and λ. It has been found that the strategy that gives fewest iterations is finding $\lambda^{(i)}$[8] first with $D(x^+)$, found using $\nabla_x l(x^+, \lambda)$, then updating $D(x^+)$, using $\nabla_x l(x^+, \lambda^{(i)})$, and then updating the multiplier once again using the new $D(x^+)$ to finally find λ^+. However, this requires an extra intermediate QR factorization which certainly cannot be reused, but is a necessary expense in our opinion. Approximate strategies for updating the QR factorization might be helpful here, if one can be devised. Other multiplier update formulas which avoid the QR factorization are also being looked into.

[8] $\lambda^{(i)}$ for 'intermediate multipliers'.

4.3 Trust region update

The strategy for updating the trust region radius δ is as follows:

Let $r = \frac{ared}{pred}$.

- **If** $r < 10^{-4}$ or $ared < 0$ (step is rejected)

Let

$$\beta_1 = \begin{cases} 0.5, & \text{if } \delta \leq 10 \\ 0.25, & \text{if } 10 < \delta \leq 10^4 \\ 0.1, & \text{if } \delta > 10^4 \end{cases};$$

then $\delta = \max\{\beta_1 \min(\delta, \|s\|), \delta_{min}\}$.

else step is accepted.

- **If** $10^{-4} \leq r < 10^{-2}, \delta = \max\{0.5\delta, \delta_{min}, \|s\|\}$

- **If** $0.01 \leq r < 0.1, \delta = \max\{0.9\delta, \delta_{min}\}$

- **If** $r \geq 0.75$,

Let

$$\beta_2 = \begin{cases} 10, & \text{if } \delta \leq 10^{-4} \\ 5, & \text{if } 10^{-4} < \delta \leq 0.1 \\ 4, & \text{if } 0.1 < \delta \leq 100 \\ 2, & \text{if } \delta > 100 \end{cases};$$

then $\delta = \min\{\beta_2\delta, \delta_{max}\}$

4.4 BFGS Approximation

The second order quantity $\nabla_x^2 l(x, \lambda)$ is estimated by the BFGS approximation (see Dennis & Schnabel [15] for details). Given the current approximation B, the updated B^+ can be described as below:

$$y = \nabla_x l(x^+, \lambda^+) - \nabla_x l(x, \lambda^+)$$

$$w = Bs$$

$$B^+ = B + \frac{yy^T}{y^T s} - \frac{ww^T}{w^T s}.$$

It can be seen easily that if B is symmetric, so is B^+. Further, if $y^T s > 0$ and B is positive definite, so is B^+.

4.5 Penalty parameter update

The penalty parameter ρ is updated according a scheme similar to that in El-Alem[4], customized for our setting.

If $pred \geq 0.5\ \rho\ (\|h(x)\|^2 - \|\nabla_x h(x)s + h(x)\|^2)$

$$\rho^+ = \rho$$

else

$$\rho^+ = 2\frac{[D^2(x)\nabla_x l(x,\lambda)]^T s + \frac{1}{2}s^T \bar{H} s}{\|h(x)\|^2 - \|\nabla_x h(x)s + h(x)\|^2} + \bar{\rho},$$

where $\bar{\rho}$ is a fixed positive parameter, chosen to be 10 in our implementation and

$$\bar{H} = D^2(x)\nabla_x^2 l(x,\lambda) + diag(\nabla_x l(x,\lambda))diag(\eta(x))$$

It is not difficult to see that this update yields a non-decreasing sequence of penalty parameters, provided $\|h(x)\|^2 - \|\nabla_x h(x)s + h(x)\|^2 > 0$.

4.6 Scaling matrix in the implementation

Vicente in [3] reports that the following form of the vector $d(x)$ is more efficient in dealing with bounds that are far from active at the solution, which is corroborated by a huge reduction in the number of iterations in our computational experience.

$$d_i(x) = \begin{cases} \min\{1, \sqrt{b_i - x_i}\}, & \text{if } (\nabla_x l(x,\lambda))_i < 0 \text{ and } b_i < \infty \\ \min\{1, \sqrt{x_i - a_i}\}, & \text{if } (\nabla_x l(x,\lambda))_i \geq 0 \text{ and } a_i > -\infty, \\ 1, & \text{otherwise} \end{cases}$$

Our computational experience suggests that the vector $\eta(x)$ should remain as defined in (3).

4.7 Stopping Criterion

The stopping criterion is partly determined by the norm of the stopping vector sv which is defined as

$$sv_i = \min\{|(\nabla_x l(x,\lambda))_i|, d_i^2(x_i)\}$$

The current point is reported as optimal when

$$\|sv\| \leq \ n*\text{stoptol} \ \text{ and } \|h\| \leq \ m*\text{htol}.$$

Our implementation uses stoptol = 10^{-6} and htol = 10^{-8}. This absolute tolerance worked well for our test problems, a relative tolerance criterion could also have been used.

4.8 Handling degeneracy

It is well known that Newton's method can face computational difficulties and convergence can be retarded if the solution is degenerate, i.e. if there exists $i \in 1, \ldots, n$ such that

$$d_i^2(x_i^*) = 0 \text{ and } (\nabla_x l(x^*, \lambda^*))_i = 0$$

To deal with the above degeneracy, $\nabla_x l(x, \lambda)$ is redefined as below in a manner similar to Coleman & Li [2] :

$$(\nabla_x l(x,\lambda))_i = \begin{cases} (\nabla_x l(x,\lambda))_i + \epsilon, & \text{if } |(\nabla_x l(x,\lambda))_i| + |d_i(x_i)| \leq |\epsilon| \\ (\nabla_x l(x,\lambda))_i, & \text{otherwise,} \end{cases}$$

where

$$\epsilon = sign((\nabla_x l(x,\lambda))_i)\mu_\epsilon$$

and, say, $\mu_\epsilon = 10^{-8}$. Note that choosing stoptol $\leq \mu_\epsilon$ can preclude convergence.

5 Observations on incorporating inequality constraints

As was stated in the introduction, this formulation can be used to handle smooth nonlinear inequality constraints by adding slack variables to the inequalities. Let us consider the general nonlinear programming problem with equalities and inequalities

$$\min_x f(x)$$

$$s.t. \ \ c(x) = 0$$

$$g(x) \leq 0$$

$$a \leq x \leq b,$$

which is equivalent to

$$\begin{aligned} \min_x \ & f(x) \\ s.t. \ \ & c(x) = 0 \\ & g(x) + s = 0 \\ & a \leq x \leq b \\ & s \geq 0. \end{aligned}$$

Let $dim(g) = p$, the number of inequalities, and $dim(c) = k$. Now the scaling matrix $D(x)$ becomes $D(x, s)$. A close look at $\nabla_{x,s} l(x, s, \lambda)$ shows that using our previous definitions, the last p components of the vector $d(x, s)$ are defined as

$$d_{n+i}(s_i) = \begin{cases} \min\{1, \sqrt{s_i}\}, & \text{if } \lambda_{n+i} \geq 0 \\ 1, & \text{otherwise} \end{cases}$$

$$\text{for } i = 1, \ldots, p.$$

A quick look at the last p components of $\nabla^2_{x,s} l(x, s, \lambda) = 0$, reveal that it is only an explicit restatement of complementarity. Observing that $\lambda_{n+i}, i = 1, \ldots, p$, are just the multipliers of the inequality constraints (commonly denoted by μ), the above definition of $d_{n+i}(s_i)$ is easily identified as the mechanism forcing nonnegativity of the Lagrange multipliers of the inequality constraints.

Finally, the requirement of $\nabla_x h(x)$ having full rank gets translated into $[\nabla_{x,s} c(x), \nabla_{x,s} g(x) + s]$ having full rank. Given that $\nabla_s c(x) = 0_{p \times k}$ and $\nabla_s g(x, s) = I_p$, it becomes equivalent to only $\nabla_x c(x)$ having full rank. Note that this requirement on the rank stems simply from the issue of $\bar{R}$ from the QR factorization being nonsingular, and not from satisfying a constraint qualification at the solution. Also note that the $m < n$ requirement earlier now becomes $p + k < n + p$, which is the same as $k < n$.

6 Some Computational Observations

- Speed of convergence is very sensitive to ν, the parameter in the normal subproblem which determines what fraction of the total trust region radius would be allowed for the horizontal subproblem. For an

efficient implementation applicable over a wide range of problems, a strategy for adapting ν based on the iteration history should certainly be devised.

- Convergence is also very sensitive to the initial trust region radius.
- It is fairly sensitive to the trust region update scheme. Here we have presented what seemed optimal based on our computational experiences.
- The number of iterations[9] is greatly reduced, often by hundreds, if we do not reject the very first step that fails to satisfy the step acceptance criterion but accept it instead.

Below are results of computational tests using a Matlab 4.2a implementation (on a SUN OS 4.1.3 workstation) on a few test problems. The best, followed by the worst performance (for the few parameter settings we attempted) are tabulated for the scaled version, unscaled versions 1, 2, & 3 and the mixed unscaled version along with their corresponding ν and δ_0. Unless otherwise mentioned, by default the initial trust region radius δ_0 is set to $2\max\{\|x_0\|, 1\}$, $\bar{\rho} = 1000, \rho_0 = 1$ and the first rejectable step is accepted. The stopping criterion is as mentioned earlier. Results from a primal-dual interior point algorithm and Matlab's active set general NLP solver *constr* are also provided for the sake of comparison.

Problem 1

$$\begin{aligned}
\min_x f(x) = x_1^2 + 3x_2 - 0.1x_3x_4 + e^{-x_2} + (x_5 - 2x_2)^2 \\
s.t.\ \ x_1 + 2x_2 + 4x_3 + 6x_4 + 7x_5 = 0 \\
x_1^2 - 3x_2^2 + 0.3x_2x_4 - x_5 = 0 \\
2x_1 + x_2 - 0.1x_5^3 = 0 \\
3x_1^2 + 4(x_2 + x_5)^2 = 25 \\
-10 \le x_i \le 10, i = 1, 2, 3, 5 \\
-11 \le x_4 \le 10
\end{aligned}$$

[9] Which is exactly the same as the number of function and gradient evaluations in our way of counting.

Three KKT points[10] were found in course of these computations:

- $[-2.2098, 1.4782, 10.0, -3.1898, -3.0868]$, with $f^* = 49.2568$ (referred to as KKT pt. 1)
- $[1.4793, -0.6858, 10.0, -9.9893, 2.8326]$ with $f^* = 29.7818$ (KKT pt. 2)
- $[0.0882, -0.73, 2.2183, 0.8134, -1.7689]$, with $f = -0.1921$, the lowest function value among the three (KKT pt. 3)

Starting point:$x_1 = -6.3, x_2 = 1.0, x_3 = 1.0, x_4 = 0.55, x_5 = 1.0$.

Subproblem type	iterations	ν	δ_0	Point of convergence
scaled	9	0.5	default	KKT pt. 1
	14	0.7	default	
unscaled v.1	40	0.5	default	KKT pt. 1
	46	0.7	default	
unscaled v.2	40	0.5	default	KKT pt. 1
	45	0.7	default	
unscaled v.3	53	0.5	default	KKT pt. 1
	69	0.7	default	
mixed unscaled	53	0.5	default	KKT pt. 1
	58	0.7	default	KKT pt. 2

The primal-dual interior point method with essentially line search globalization [11] converged in 31 iterations to KKT point 2. The stopping criterion for the primal dual algorithm was

$$\|\nabla_x l(x, \lambda, \mu_a, \mu_b)\| + \|h\| \leq 10^{-6}$$

Matlab's *constr* converged in 9 iterations to KKT point 1.

Starting point of x_1 was changed to $+6.3$.

[10] Karush-Kuhn Tucker points, i.e., points satisfying first order necessary conditions for optimality.

[11] The implementation we used might not be the best available.

Subproblem type	iterations	ν	δ_0	Point of convergence
scaled	12	0.6	default	KKT pt. 2
	81	0.45	1.0	
unscaled v.1	29	0.8	1.0	KKT pt. 3
	120	0.8	default	KKT pt. 2
unscaled v.2	29	0.8	1.0	KKT pt. 3
	121	0.8	default	KKT pt. 2
unscaled v.3	34	0.7	1.0	KKT pt. 3
	108	0.75	default	KKT pt. 2
mixed unscaled	29	0.8	default	KKT pt. 3
	134	0.6	default	KKT pt. 2

Primal-dual interior point, as before, converged in 31 iterations to the second KKT point.
Matlab's *constr* did not converge in 500 iterations.

Problem 2

Same as the first problem, except that the fourth equality was now converted to an inequality, i.e.,

$$3x_1^2 + 4(x_2 + x_5)^2 \leq 25$$

and a slack variable was added to get it in our form:

$$3x_1^2 + 4(x_2 + x_5)^2 + x_6 = 25$$

$$x_6 \geq 0.$$

Observe that the previous KKT points are still KKT points for this problem (with the slack variable $x_6 = 0.$) so they shall be referred to as KKT points 1, 2 and 3 as before.

Starting point: [6.3, 1, 1, 0.55, 1, 1] (last variable is a slack):

Subproblem type	iterations	ν	δ_0	Point of convergence
scaled	31	0.6	1.0	KKT pt. 2
	> 500	0.65	default	n/a
unscaled v.1	46	0.8	1.0	KKT pt. 3
	95	0.65	default	KKT pt. 2
unscaled v.2	43	0.8	1.0	KKT pt. 3
	99	0.65	default	KKT pt. 2
unscaled v.3	148	0.5	100	KKT pt. 3
	256	0.65	1.0	KKT pt. 2
mixed unscaled	55	0.65	default	KKT pt. 3
	> 500[a]	0.6	default	n/a

[a] Converges in 49 iterations to KKT pt. 3 if the second merit function is dropped.

Primal-dual interior point converges in 48 iterations to the second KKT point.

Matlab's *constr* converges in 101 iterations to point 3.

Starting point changed to $[-9, -9, -9, -9, -9, 1]$:

Subproblem type	iterations	ν	δ_0	Point of convergence
scaled	50	0.8	1.0	KKT pt. 3
	141	0.65	default	
unscaled v.1	100	0.6	default	KKT pt. 3
	119	0.65	default	
unscaled v.2	98	0.6	default	KKT pt. 3
	117	0.65	default	
unscaled v.3	103	0.8	default	KKT pt. 3
	119	0.5	default	
mixed unscaled	104	0.6	default	KKT pt. 3
	129	0.65	default	

Primal-dual interior point does not converge in 800 iterations.

Matlab takes 22 iterations to converge to point 3.

Starting point changed to $[9.5, 9.5, 9.5, 9.5, 9.5, 1]$:

Subproblem type	iterations	ν	δ_0	Point of convergence
scaled	39	0.7	1.0	KKT pt. 2
	79	0.8	1.0	
unscaled v.1	103	0.8	1.0	KKT pt. 3
	125	0.65	default	
unscaled v.2	140	0.8	default	KKT pt. 2
	183	0.8	1.0	KKT pt 3
unscaled v.3	69	0.8	default	KKT pt. 3
	87	0.65	default	
mixed unscaled	69	0.8	default	KKT pt. 3
	87	0.65	default	

Primal-dual converges in 68 iterations to the second KKT point.

Matlab converges to KKT point 2 in 74 iterations.

Problem 3

The last three equalities in problem 1 were converted to $\leq$ inequalities and slacks were added to them.

All the convergent runs of our algorithms converged to only one KKT point: $[-0.0131, -0.8609, 1.6511, 1.1007, -1.6390, 0.8687, 0.4467, 0]$ (last three slack variables) with $f = -0.3921$, hence the point of convergence is omitted from the table. The first two inequalities are inactive and the last one is active.

Starting point: [2, 6, 6, -6, -6, 2, 1, 1] (last three variables are slacks on the inequalities)

Subproblem type	iterations	ν	δ_0
scaled	> 500	all[a]	def., 1.0
unscaled v.1	187	0.8	default
	224	0.65	1.0
unscaled v.2	202	0.75	1.0
	220	0.65	default
unscaled v.3	> 500	all	n/a
mixed unscaled	78	0.7	1.0
	223	0.65	default

[a]Indicates that it did not converge for all the parameter settings attempted

Primal-dual interior point did not converge in 500 iterations.

Matlab converged in 38 iterations.

Starting point changed to $[6.3, 1, 1, 0.55, 1, 2, 1, 1]$.

Subproblem type	iterations	ν	δ_0
scaled	> 500	all	def., 1.0
unscaled v.1	167	0.8	default
	175	0.65	default
unscaled v.2	166	0.65	default
	> 500	0.8	default
unscaled v.3	191	0.65	default
	not conv.	other[a]	
mixed unscaled	82	0.55	10.0
	266	0.65	1.0

[a]Did not converge for other attempted parameter settings

Primal-dual interior point failed to converge within 500 iterations.

Matlab converged in 60 iterations.

Problem 4

(Hock & Schittkowski test problem set, no. 100)

$$\min_x f(x)$$

$$= (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$$

$$s.t.\ \ 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 127$$

$$7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 282$$

$$23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 196$$

$$4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

This problem was converted into our required form using slack variables, and $f(x)$ was scaled by 10^{-3} to reduce ill-conditioning.

Convergent runs converged to only one KKT point,

$$[2.3309, 1.9459, -0.4747, 4.3794, -0.6238, 1.0372, 1.5954, 0, 252.5904, 144.9098, 0]$$

with $f^* = 680.6$, the first and last inequalities being active, and the second and third being 'highly inactive'. Instead of focussing on the difference between the best and worst number of iterations due to changes in δ_0 and ν, we tabulate representative results for each of the algorithms below.

Starting point: $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$ (last four slacks)

Subproblem type	No. of iterations	ν	δ_0
scaled	$> 500^a$		
unscaled v.1	238	0.6	default
unscaled v.2	229	0.6	default
unscaled v.3	ill-cond		
mixed unscaled	268	0.55	default

[a] high ill-conditioning after 300 iterations

The primal-dual interior point code available to us required at least one upper bound, so we set the upper bound on x_2 to 10000 (which is not active at the solution), however it did not converge within 500 iterations.

Matlab's *constr* converged in 273 iterations.

Problem 5

(Hock & Schittkowski test problem set, no. 81)

$$\min_x e^{x_1x_2x_3x_4x_5} - \frac{1}{2}(x_1^3 + x_2^3 + 1)^2$$

$$
\begin{aligned}
s.t. \quad & \|x\|^2 = 10 \\
& x_2 x_3 - 5 x_4 x_5 = 0 \\
& x_1^3 + x_2^3 + 1 = 0 \\
& -2.3 \leq x_1, x_2 \leq 2.3 \\
& -3.2 \leq x_3, x_4, x_5 \leq 3.2
\end{aligned}
$$

Eleven distinct KKT points were found in course of the numerical experiments, most of them with $f^* = 1$, one with $f^* = 0.0539$ and another with $f^* = 0.4389$. For the sake of elegance, a list of those KKT points and thereof is skipped.

Starting point: $[1, 1, 1, 1, 1]$

Subproblem type	No. of iterations	ν	δ_0
scaled	22	0.65	default
unscaled v.1	39	0.75	default
unscaled v.2	35	0.75	default
unscaled v.3	42	0.65	default
mixed unscaled	33	0.75	default

Primal-dual converges in 49 iterations.

Matlab converges in 96 iterations.

New starting point: $[2, -2, 2, -2, 2]$

Subproblem type	No. of iterations	ν	δ_0
scaled	363	0.8	default
unscaled v.1	306	0.65	default
unscaled v.2	19	0.65	default
unscaled v.3	77	0.75	1.0
mixed unscaled	42	0.65	1.0

Primal-dual interior point does not converge in 500 iterations, and neither does Matlab's *constr*.

6.1 Inferences

Since the range of problems tested here was fairly restricted, no attempts will be made to make sweeping generalizations based on the above observations. It does seem however that the scaled version fails to perform well in the presence of nonlinear equalities, though it does remarkably well otherwise. The sensitivities of the convergence to ν and δ_0 were among the two chief things we aimed to and did demonstrate. The algorithms are still under development and further testing is in progress. The mixed unscaled version is arguably the most robust of the algorithms and will be the prime focus of further development. Our forthcoming implementations in Fortran and C will use DGQTPAR to solve the subproblems, and could provide further improvements on the performance reported here, since this implementation only uses dogleg steps.

6.2 Advantages

The chief advantages of this algorithm are over other algorithms for the general NLP are as follows:

- *Solves the 'sticking problem':* Many primal-dual interior point algorithms have the undesirable property that some of the bounded variables often get 'attracted' to the wrong bounds and when the iterates reach the bounds they 'stick' to those bounds which do not correspond to the solution, and hence convergence is hindered.[12] This is one problem that is apparently solved by the Coleman-Li scaling, since the scaled step is sufficiently 'angled away' from approaching bounds. The starting points for several of our tests were very close to the variable bounds, but the 'sticking phenomenon' never occurred. Similar observations are also made in Coleman & Li [1], [2] and in Vicente [3].

- *Reduces the number of variables:* Since this algorithm never computes or requires the multipliers of the bound constraints, the number of variables is greatly reduced compared to primal-dual interior point methods or active set algorithms.

- *Advantage over active set algorithms:* This algorithm wins over active set algorithms in the same manner as any other interior point algorithm. In particular, it is well-known that active set strategies can be

[12] the reason for this becomes clear on examining the Newton step on the complementarity conditions.

very inefficient in the presence of a large number of bounds and/or a large number of inequality constraints, especially when most of the bounds or inequality constraints are not active. Further, an active set strategy can identify the wrong active set and converge to a point where some of the inequalities or bounds are not satisfied.

7 Conclusion

An interior point algorithm for solving the general nonlinear programming problem using trust region globalization was presented. It is quite likely that it would be possible to rigorously prove under reasonable assumptions global convergence of the iterates and local q-quadratic convergence in the presence of exact second-order information or local q-superlinear convergence using a secant aproximation. Our aim in this paper, in addition to motivating theoretical analyses of this algorithm, was to extract from computational experiments ingredients, perhaps heuristics, needed to expedite the semi-local convergence of this algorithm. With more computational tests on a wider set of problems in progress, the author hopes that this treatise would attract other computational scientists and theoreticians to study this algorithm in further depth. In fact, Dr. Mahmoud El-Alem is currently involved in proving convergence for these algorithms.

8 Appendix A: Non-equivalence of subproblems in scaled and unscaled steps

Consider Coleman & Li's scaled and unscaled subproblems in [1], rewritten in our notation (B is an approximation to $\nabla_x^2 f(x)$, $D = D(x)$, and $J = diag(\nabla_x f(x))diag(\eta(x))$:

Scaled subproblem

$$\min_{\hat{s}} \frac{1}{2}\hat{s}^T[DBD + J]\hat{s} + [D\nabla_x f(x)]^T\hat{s}$$
$$s.t. \quad \|\hat{s}\| \leq \delta$$

Unscaled subproblem

$$\min_{s} \frac{1}{2}s^T(B + JD^{-2})s + \nabla_x f(x)^T s$$

$$s.t. \quad \|D^{-1}s\| \leq \delta.$$

Let the multipliers of the trust-region inequalities for the two subproblems be respectively $\hat{\mu}$ and μ. Suppose the Newton steps are outside the trust regions, so that corresponding to the optimal steps, $\mu > 0$ and $\hat{\mu} > 0$ and the trust region inequalities are active. Suppose there does not exist an eigenvalue of D^2 equal to $\frac{\mu}{\hat{\mu}}$ that has s as its corresponding eigenvector. Then

$$D\hat{s} \neq s$$

Proof:
Recalling the necessary condition for optimality, the optimal step from the scaled subproblem must satisfy

$$(DBD + J + \hat{\mu}I)\hat{s} = -D\nabla_x f(x)$$
$$\equiv (DB + JD^{-1} + \hat{\mu}D^{-1})D\hat{s} = -D\nabla_x f(x).$$

Since J, D are diagonal, we have

$$(B + JD^{-2} + \hat{\mu}D^{-2})D\hat{s} = -\nabla_x f(x). \tag{9}$$

Similarly, the optimal step from the unscaled subproblem satisfies

$$\equiv (B + JD^{-2} + \mu)s = -\nabla_x f(x). \tag{10}$$

The proof now follows by contradiction. Assume that $D\hat{s} = s$. Then substituting s for $D\hat{s}$ in (9) and subtracting (9) from (10) we get

$$(\mu - \hat{\mu}D^{-2})s = 0.$$

Thus, since $\mu > 0$

$$D^2 s = \frac{\hat{\mu}}{\mu}s.$$

But the above cannot hold since, by hypothesis, $\frac{\mu}{\hat{\mu}}$ is not an eigenvalue of D^2 with corresponding eigenvector s.

Hence, by contradiction,

$$D\hat{s} \neq s \qquad (QED)$$

It is noteworthy that since D^2 is diagonal with nonzero entries, its eigenvectors are the canonical basis vectors e_i of $\mathcal{R}^n$, hence for a step s to even qualify as an eigenvector of D^2 would mean that it must have only one nonzero element.

9 Acknowledgements

The author would like to thank his advisor, Dr. John Dennis for his helpful guidance and Dr. Mahmoud El-Alem for carefully reading the report several times and offering numerous helpful suggestions, including pointing out a missing term in the tangential subproblem in an earlier version of this report. Dr. El-Alem and the author are currently involved in further developing this algorithm, particularly using a reduced space approach.

References

[1] Thomas F. Coleman and Yuying Li. *An Interior Trust Region Approach For Nonlinear Minimization Subject to Bounds.* SIAM J. Optimization, Vol. 6, No. 2, pp. 418-445, May 1996, pp 418-445.

[2] Thomas F. Coleman and Yuying Li. *On the Convergence of Interior-reflective Newton Methods for Nonlinear Minimization Subject to Bounds.* Mathematical Programming, 67, 1994, pp. 189-224.

[3] Luís N. Vicente. *Trust Region Interior-Point Algorithms for a Class of Nonlinear Programming Problems.* Ph.D. thesis, Dept. of Computational and Applied Mathematics, Rice University, Houston, TX 77251, March 1996.

[4] Mahmoud El-Alem. *A global convergence theory for the Celis-Dennis-Tapia trust-region algorithm for constrained optimization.* SIAM J. Numerical Analysis, 28, 1991, pp. 266-290.

[5] Mahmoud El-Alem. *Convergence to a Second-order Point for a Trust-region Algorithm with a Nonmonotonic Penalty Parameter for Constrained Optimization.* Tech. Report TR95-28. Dept. of Computational and Applied Mathematics, Rice University, Houston, TX 77251, 1995.

[6] Mahmoud El-Alem. *A Robust Trust-region Algorithm with a Nonmonotonic Penalty Parameter Scheme for Constrained Optimization.* SIAM J. Optimization, 5, 1995, pp. 348-378.

[7] J. Goodman. *Newton's Method for Constrained Optimization.* Mathematical Programming, 33, 1985, pp. 162-171.

[8] M. Celis, J. E. Dennis and R. A. Tapia. *A Trust-region Strategy for Nonlinear Equality Constrained Optimization.* Numerical Optimization, 1984. SIAM, Philadelphia, Pennsylvania, 1985, pp. 71-82.

[9] J. J. Moré and D. C. Sorensen. *Computing a Trust Region Step.* SIAM J. Scientific and Statistical Computing, 4, 1983, pp. 553-572.

[10] N. Alexandrov. *Multilevel Algorithms for Nonliear Equations and Equality Constrained Optimization.* Ph.D. thesis, Dept. of Computational and Applied Mathematics, Rice University, Houston, TX 77251, May 1993.

[11] Y. Zhang. *Computing a Celis-Dennis-Tapia trust-region step for equality constrained optimization.* Math. Programming, 55, 1992, pp. 109-124.

[12] S. A. Santos and D. C. Sorensen. *A new matrix-free algorithm for the large-scale trust-region subproblem.* Tech. report TR95-20, Dept. of Computational and Applied Mathematics, Rice University, 1994.

[13] D. C. Sorensen. *Minimization of a large scale quadratic function subject to an ellipsoidal constraint.* Tech. Report TR94-27, Dept. of Computational and Applied Mathematics, Rice University, Houston, TX 77251, 1994.

[14] T. Steihaug. *The conjugate gradient method and trust regions in large scale optimization.* SIAM J. Numerical Analysis, 20, 1983, pp. 626-637.

[15] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM, Philadelphia, 1996. Originally published by Prentice Hall, Englewood Cliffs, New Jersey, 1983.